**Theorem 1** If the input graph to FR is acyclic, then the output is acyclic and destination-oriented.

**Proof:** Lemma 3 states that FR terminates, so there is a final graph in which at most one vertex, \( D \), is a sink. By Lemma 4, if input graph is acyclic, so is the final graph. By Lemma 1, \( D \) is a sink, so by Lemma 2, the graph is \( D \)-oriented.

---

1 Partial Reversal Algorithm (PR)

**Idea:** Set of links to be reversed is computed as follows. For each sink in chosen set \( S \) of sinks, link \((u, v)\) is reversed by \( v \) iff the link has not been reversed by \( u \) since the last iteration in which \( v \) took a step.

1.1 Algorithm - Partial Reversal (PR)

**Require:** Directed graph \( \tilde{G} = (V, \tilde{E}) \) with distinguished vertex \( D \).

1. for each \( v \in V \) do
2. \( \text{list}[v] \leftarrow \emptyset \)
3. end for
4. while \( \tilde{G} \) has a sink other than \( D \) do
5. choose some non-empty subset \( S \) of sinks in \( \tilde{G} \) s.t. \( D \not\in S \)
6. for all \( v \in S \) do
7. if \( N_{\tilde{G}}(v) \neq \text{list}[v] \) then
8. reverse the direction of all links incident on \( v \) whose source is not in \( \text{list}[v] \)
9. else
10. reverse the direction of all links incident on \( v \)
11. end if
12. for all \( u \) such that \((u, v)\) was just reversed do
13. add \( v \) to \( \text{list}[u] \)
14. end for
15. \( \text{list}[v] \leftarrow \emptyset \)
16. end for
17. end while

**Lemma 2** PR terminates.

**Proof:** Suppose, for contradiction, there is some execution of PR that does not terminate. Let \( W \) be the set of vertices that are chosen infinitely often in \( S \). Since \( V - \text{finite} \Rightarrow W \neq \emptyset \). Also, \( D \in V - W \Rightarrow V - W \neq \emptyset \).

Since \( G \) is connected, \((\exists u \in W, v \in V - W) (u, v) \in E \). Let \( t \) be the iteration at which \( v \) takes its last step (if \( v \) takes no steps, then \( t = 0 \)). We claim that vertex \( u \) reverses \((v, u)\) in either the first or the second step it takes after iteration \( t \).
Suppose \( u \) does not reverse \((v, u)\) in its first step after iteration \( t \). At the end of this step, \( \text{list}[u] \) is emptied. Since \( v \) takes no more steps, \( v \) is never later added to \( \text{list}[u] \).

Then, when \( u \) takes its second step after iteration \( t \), \( v \) is not in \( \text{list}[u] \), so the link \((v, u)\) is reversed to be directed from \( u \) to \( v \).

Thus, after at most 2 steps by \( u \) after iteration \( t \), edge \((u, v)\) remains directed away from \( u \), so \( u \) never becomes a sink again, so takes no more steps, a contradiction.

Consider the following example of PR algorithm:

In comparison, FR algorithm for the same problem is the following:

Remark 3 PR preserves acyclicity, but this result is harder to prove, and the proof is omitted here.

2 Vertex Labels

To implement GLR, Gafni&Bertsekas proposed using vertex labels, or heights. The labels are chosen from a totally ordered set. The vertex labels are used to induce an orientation \( \vec{G} \) of an undirected graph (link between \( u \) and \( v \) is directed from \( u \) to \( v \) \((u \rightarrow v)\) iff the label of \( u \) is greater than the label of \( v \)).

Therefore, a sink is a vertex whose label (height) is a local minimum.

Remark 4 It’s implied here that every vertex knows the labels of its neighbors.

Definition 1 Let \( L \) be a totally ordered, countably infinite set of labels. We partition it into a disjoint collection \( \{L_v | v \in V\} \) of subsets of labels for each vertex (for example, ordered pairs with a node name as an element of a pair) such that \((\forall a \in L) (\forall v \in V) (\exists a_v \in L_v) a < a_v\).

The last condition in definition 1 guarantees that each \( L_v \) is unbounded in \( L \). Without such a condition there might be a situation when we will be unable to reverse an incoming link to a node due to inability to pick big enough label for that node.

Since the \( L_v \)'s are disjoint, no two vertices have the same label, so the direction of edges is always well-defined.